# Assignment by Malakounidis Paris

Github repo: <a href="https://github.com/m3fisto/assignment">https://github.com/m3fisto/assignment</a>

The project was developed on a macbook Pro using the VScode IDE and python 3.9. It was also tested on a Win10 pc with python 3.11.

The programming language is Python as requested in the exercise. Flask is used to create a simple API service and Pytest is used for both the functional and the non functional testing.

All the needed python dependencies can be found in requirements.txt located on the root folder of the project. Docker is also used for containerizing the service

## The service

A REST api server is created using Flask. It serves GET requests on 3 endpoints on <a href="http://localhost:8800">http://localhost:8800</a> The available endpoints are:

/people/<num>

/planets/<num>

/starships/<num>

Once the service is up it serves predefined json files

A server log file is created with the startup of the service on the root folder of the project and is named server\_app.log. The log keeps track of HTTP requests on the server and the relevant response status code.

## **Functional Tests**

Functional tests were created in order to test happy and edge scenarios on the service. They are located in the /tests folder and named service\_test.py A sample manual execution of the tests is the below(given the app is running):

pytest ./Tests/service\_test.py --html=reportFT.html
and outputs:

```
Darismalakounidis@@2CG1QAWDBR python_assignment % pytest ./test/service_test.py —html=reportFT.html

platform darvin — Python 3.9.16, pytest-7.1.2, pluggy-1.0.0
benchmark: 4.0.0 (defaults: timer=time.perf_counter disable_gc=False min_rounds=5 min_time=0.000005 max_time=1.0 calibration_precision=10 warmup=False warmup_iterations=1
000000)
routdir: /Users/parismalakounidis/Desktop/projects/python_assignment, configifile: pytest.ini
plugins: reporter—html=0.8.2, playwright=0.3.0, html=3.2.0, base=url=2.0.0, docker=1.0.1, reporter=0.5.2, metadata=2.0.4, benchmark=4.0.0, flask=1.2.0
collected 7 items

test/service_test.py::test_get_people_PASSED

test/service_test.py::test_get_people_error PASSED

test/service_test.py::test_get_planet_PASSED

test/service_test.py::test_get_planet_error PASSED

test/service_test.py::test_get_starships PASSED

test/service_test.py::test_get_starships_error PASSED

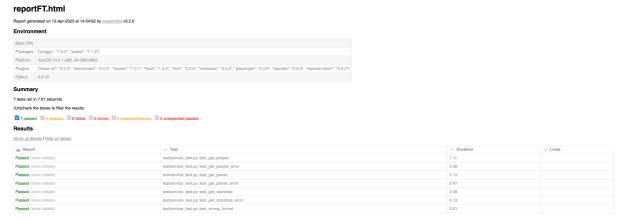
test/service_test.py::test_get_starships_error PASSED

test/service_test.py::test_get_starships_error PASSED

[38%]
test/service_test.py::test_get_planet_PASSED

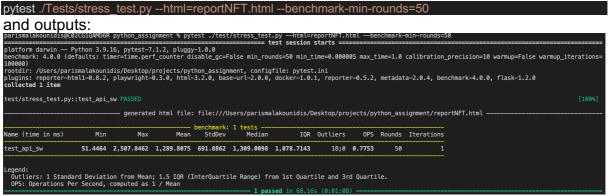
[38%]
test/service_test.py::test_get_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_passed_p
```

An .html report is also created with filename reportFT.html:



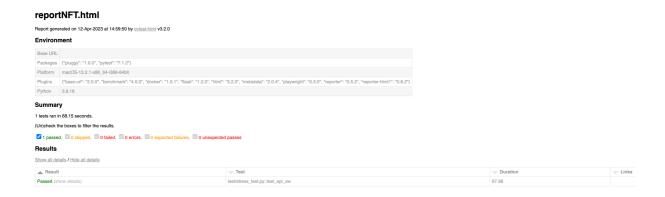
## **Performance Tests**

Performance tests were created in order to measure the average response time of the server. They are located in the /tests folder and named stress\_test.py A random delay has been introduced in the server (between 0-2,5 secs per request) Sample manual execution of the tests is the following (given the service is up and running:



On the above printout we can identify the Mean and standard deviation values. The test run for approximately one minute and the mean response time was 1.28 sec which is expected since the incurred delay on the server side is a random [0:2.5] seconds.

An .html report is also created with filename reportNFT.html:



## Docker

A local docker image has to be created so the suite can run automatically. This can be done by executing on the root folder (where the dockerfile is located):

### docker build -t api-sw

This will create an image named api-sw. This image contains the API service

The image can be started manually via:

docker run -d --name api-sw -p 8800:8800 api-sw

This will create a container that listens on localhost:8800 and serves the app

# Test Suite execution

Given the prerequisite that the docker image has been created, we are able to run the tests with one command. This can be achieved with the pytest-docker plugin which spawns the container in order to run the tests and then terminates it in an approach similar to Ephemeral Environments

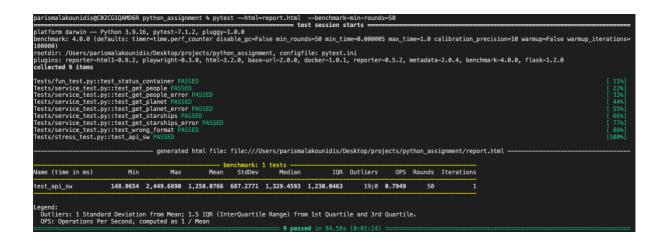
The above can be done with (the app must not be running either manually or via docker):

# pytest --html=report.html --benchmark-min-rounds=50

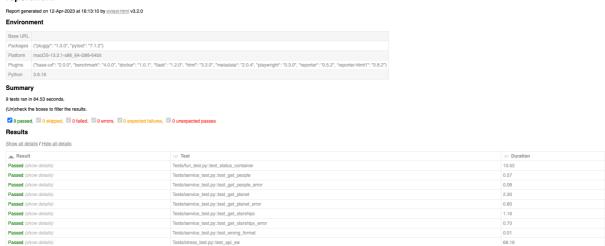
This will perform the following actions:

- Start the api-sw container
- Wait until it is reachable
- Run the functional tests
- Run the performance tests
- Stop and delete the container

and outputs:



# A combined html report was created with filename report.html:



# Windows10 execution

- Install docker desktop
- install python (eg 3.10)
- install pip if needed
- open terminal in project root folder
- docker build -t api-sw .
- pip install pytest requests pytest-docker pytest-benchmark pytest-html
- pytest --html=report.html --benchmark-min-rounds=50

console output in Windows PowerShell:

```
Distribution of the second of 
   ollected 9 items
ests/functest.py::test_status_container PASSED
ests/service_test.py::test_get_people PASSED
ests/service_test.py::test_get_people_error PASSED
ests/service_test.py::test_get.pole_error PASSED
ests/service_test.py::test_get.planet PASSED
ests/service_test.py::test_get.planet.perror PASSED
ests/service_test.py::test_get.starships_error PASSED
ests/service_test.py::test_get.starships_error PASSED
ests/service_test.py::test_get.starships_error PASSED
ests/service_test.py::test_get.starships_error PASSED
ests/stress_test.py::test_apin_stress_test.ps.starships_error PASSED
ests/stress_test.py::test_apin_stress_test.ps.starships_error PASSED
benchmark: 1 tests
Name (time in ms) Min Max Mean StdDev Median IQR Outliers OPS Rounds Iterations
test_api_sw 95.9487 2,501.9686 1,146.4119 721.5775 1,084.3380 1,209.4511 22;0 0.8723 50 1
 egend:
Outliers: 1 Standard Deviation from Mean: 1.5 IQR (Interquartile Range) from 1st Quartile and 3rd Quartile.
OPS: Operations Per Second, computed as 1 / Mean
OPS: Operations Per Second, computed as 2 / Mean
```

# Html report:

### report.html

Report generated on 12-Apr-2023 at 18:42:54 by pylest-html v3:2.0

Padages [plugg/"1.0.0", pytest". 73.0"]
Platform Windows-10-10.0.1904-8-P0
Plugins ("benchmark", "4.0.0", "docker", "1.0.1", "html", "3.2.0", "metadata", "2.0.4")
Python 3.11.3

9 tests ran in 82.81 seconds.

(Un)check the boxes to filter the results.

✓ 9 passed, 
☐ 0 skipped, ☐ 0 failed, ☐ 0 errors, ☐ 0 expected failures, ☐ 0 unexpected passes

▲ Result	Test	- Duration
Passed (show details)	Tests/fun_test.py::test_status_container	11.38
Passed (show defails)	Tests/service_test.py::test_get_people	1.27
Passed (show details)	Tests/service_test.pg:test_get_people_error	1.13
Passed (show details)	Tests/service_test.py::test_get_planet	1.38
Passed (show details)	Tests/service_test.py::test_get_planet_error	2.45
Passed (show details)	Tests/service_test_get_starships	0.34
Passed (show details)	Tests/service_test_get_starships_error	0.76
Passed (show details)	Tests/service_test.py:itest_wrong_format	0.00
Passed (show details)	Tests/stress_test.py::test_api_sw	63.39